Listing of Claims:

1. (currently amended) An apparatus comprising:

a key generator to generate an operating system nub key (OSNK) unique to an operating system (OS) nub, the OS nub being part of an operating system to run on a platform comprising a processor capable of operating in an isolated execution mode in a ring 0 operating mode, wherein the processor also supports one or more higher ring operating modes, as well as a normal execution mode in at least the ring 0 operating mode; and

a usage protector coupled to the key generator to protect usage of a subset of a software environment using the OSNK;

the key generator to generate the OSNK based at least in part on a master binding key (BK0) of the platform and an identification of the OS nub;

wherein the usage protector performs at least one operation selected from the group consisting of:

encrypting a value while operating in isolated execution mode; and

decrypting an encrypted value while operating in isolated execution mode.

2. (canceled)

3. (currently amended) The apparatus of ~~claim 2~~ claim 1, wherein the identification comprises a hash value of at least one item selected from the group consisting of the OS nub and a certificate representing the OS nub.

4. (original) The apparatus of claim 1 wherein the usage protector comprises:

an encryptor to encrypt the subset of the software environment using the OSNK, the encrypted subset being stored in a storage; and

a decryptor to decrypt the encrypted subset using the OSNK, the encrypted subset being retrieved from the storage.

5. (original) The apparatus of claim 1 wherein the usage protector comprises:

an encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

a decryptor to decrypt the encrypted first hash value using the OSNK, the encrypted first hash value being retrieved from the storage; and

a comparator to compare the decrypted first hash value to a second hash value to generate a compared result, the compared result indicating whether the subset of the software environment has been modified.

6. (original) The apparatus of claim 1 wherein the usage protector comprises:

a first encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

a second encryptor to encrypt a second hash value using the OSNK; and

a comparator to compare the encrypted second hash value to the encrypted first hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

7. (original) The apparatus of claim 1 wherein the usage protector comprises:

a decryptor to decrypt a protected private key to generate a private key using the OSNK;

a signature generator coupled to the decryptor to generate a signature of the subset of the software environment using the private key, the signature being stored in a storage; and

a signature verifier to verify the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not modified flag indicating whether the subset has been modified.

8. (original) The apparatus of claim 1 wherein the usage protector comprises:

a manifest generator to generate a manifest of the subset of the software environment, the manifest describing the subset of the software environment, the manifest being stored in a storage;

a signature generator coupled to the manifest generator to generate a manifest signature using a private key, the private key being decrypted by a decryptor using the OSNK, the manifest signature being stored in the storage;

a signature verifier to verify the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and

a manifest verifier to verify the manifest to generate a manifest verified flag, the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested at a test center, the test center generating a pass/fail signal to indicate whether the subset has been modified.

9. (canceled)

10. (original) The apparatus of claim 1 wherein the software environment comprises an operating system selected from the group consisting of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

11. (previously presented) The apparatus of claim 1 wherein the subset of the software environment comprises a registry of an operating system.

12. (currently amended) The apparatus of ~~claim 2~~ claim 1, wherein the BK0 is generated at random on a first invocation of a processor nub.

13. (currently amended) A method comprising:

generating an operating system nub key (OSNK) unique to an operating system (OS) nub, the OS nub being part of an operating system to run in a software environment on a platform comprising a processor capable of operating in an isolated execution mode in a ring 0 operating mode, wherein the processor also supports one or more higher ring operating modes, as well as a normal execution mode in at least the ring 0 operating mode; and

protecting usage of a subset of the software environment using the OSNK;

wherein the operation of protecting usage of a subset of the software environment comprises at least one operation selected from the group consisting of:

encrypting a value while operating in isolated execution mode; and

decrypting an encrypted value while operating in isolated execution mode; and

wherein the operation of generating an OSNK comprises generating the OSNK based at least in part on an identification of the OS nub and a master binding key (BK0) of the platform.

14. (canceled)

15. (currently amended) The method of ~~claim 14~~ claim 13, wherein the identification comprises a hash value of at least one item selected from the group consisting of the OS nub and a certificate representing the OS nub.

16. (original) The method of claim 13 wherein protecting usage comprises:

encrypting the subset of the software environment using the OSNK;

storing the encrypted subset in a storage; and

decrypting the encrypted subset from the storage using the OSNK.

17. (original) The method of claim 13 wherein protecting usage comprises:

encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

decrypting the encrypted first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being retrieved from the storage; and

comparing the decrypted first hash value to a second hash value to generate a compared result, the decrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

18. (original) The method of claim 13 wherein protecting usage comprises:

encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

encrypting a second hash value using the OSNK; and

comparing the encrypted first hash value to the encrypted second hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

19. (original) The method of claim 13 wherein protecting usage comprises:

decrypting a protected private key to generate a private key using the OSNK;

generating a signature of the subset of the software environment using the private key, the signature being stored in a storage; and

verifying the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not modified flag indicating whether the subset of the software environment has been modified.

20. (original) The method of claim 13 wherein detecting comprises:

generating a manifest of the subset of the software environment, the manifest describing the subset of the software environment, the manifest being stored in a storage;

generating a manifest signature of the manifest using a private key, the private key being decrypted using the OSNK, the manifest signature being stored in the storage;

verifying the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and

verifying the manifest to generate a manifest verified flag, the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested at a test center, the test center generating a pass/fail signal, the pass/fail signal indicating whether the subset of the software environment has been modified.

21. (canceled)

22. (original) The method of claim 13 wherein the software environment comprises an operating system selected from the group consisting of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

23. (previously presented) The method of claim 13 wherein the subset of the software environment comprises a registry of the operating system.

24. (currently amended) The method of ~~claim 14~~ claim 13, wherein the BK0 is generated at random on a first invocation of a processor nub.

25. (currently amended) A computer program product comprising:

a computer usable medium having computer program code embodied therein, the computer program product having:

computer readable program code to generate an operating system nub key (OSNK) unique to an operating system (OS) nub, the OS nub being part of an operating system to run in a software environment on a platform comprising a processor capable of operating in an isolated execution mode in a ring 0 operating mode, wherein the processor also supports one or more higher ring operating modes, as well as a normal execution mode in at least the ring 0 operating mode; and

computer readable program code to protect usage of a subset of the software environment using the OSNK;

wherein the computer readable program code to generate the OSNK comprises computer readable program code to generate the OSNK based at least in part on an identification of the OS nub and a master binding key (BK0) of the platform; and

wherein the operation of protecting usage of a subset of the software environment comprises at least one operation selected from the group consisting of:

encrypting a value while operating in isolated execution mode; and

decrypting an encrypted value while operating in isolated execution mode.

26. (canceled)

27. (currently amended) The computer program product of ~~claim 26~~ claim 25, wherein the identification comprises a hash value of at least one item selected from the group consisting of the OS nub and a certificate representing the OS nub.

28. (original) The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for encrypting the subset of the software environment using the OSNK;

computer readable program code for storing the encrypted subset; and

computer readable program code for decrypting the encrypted subset from the storage using the OSNK.

29. (original) The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

computer readable program code for decrypting the encrypted first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being retrieved from the storage; and

computer readable program code for comparing the decrypted first hash value to a second hash value to generate a compared result, the decrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

30. (original) The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for encrypting a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

computer readable program code for encrypting a second hash value using the OSNK; and

computer readable program code for comparing the encrypted first hash value to the encrypted second hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

31. (original) The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for decrypting a protected private key to generate a private key using the OSNK;

computer readable program code for generating a signature of the subset of the software environment using the private key, the signature being stored in a storage; and

computer readable program code for verifying the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not modified flag indicating whether the software environment has been modified.

32. (original) The computer program product of claim 25 wherein the computer readable program code for protecting usage comprises:

computer readable program code for generating a manifest of the subset of the software environment, the manifest being stored in a storage;

computer readable program code for generating a manifest signature of the manifest using a private key, the private key being decrypted using the OSNK, the manifest signature being stored in the storage;

computer readable program code for verifying the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and

computer readable program code for verifying the manifest to generate a manifest verified flag , the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested at a test center, the test center generating a pass/fail signal, the pass/fail signal indicating whether the subset of the software environment has been modified.

33. (canceled)

34. (original) The computer program product of claim 25 wherein the software environment comprises an operating system selected from the group consisting of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

35. (previously presented) The computer program product of claim 25 wherein the subset of the software environment comprises a registry of an operating system.

36. (currently amended) The computer program product of ~~claim 26~~ claim 25, wherein the BK0 is generated at random on a first invocation of a processor nub.

37. (currently amended) A system comprising:

a processor capable of operating in an isolated execution mode in a ring 0 operating mode, wherein the processor also supports one or more higher ring operating modes, as well as a normal execution mode in at least the ring 0 operating mode;

storage response to the processor, the storage storing at least a subset of a software environment to run on the system:

an operating system (OS) nub;

a key generator to generate an operating system nub key (OSNK) unique to the OS nub, based at least in part on an identification of the OS nub and a master binding key (BK0) of the system; and

a usage protector coupled to the key generator to protect usage of a subset of the software environment using the OSNK;

wherein the operation of protecting usage of a subset of the software environment comprises at least one operation selected from the group consisting of:

encrypting a value while operating in isolated execution mode; and

decrypting an encrypted value while operating in isolated execution mode.

38. (canceled)

39. (currently amended) The system of ~~claim 38~~ claim 37, wherein the identification comprises a hash value of at least one item selected from the group consisting of the OS nub and a certificate representing the OS nub.

40. (original) The system of claim 37 wherein the usage protector comprises:

an encryptor to encrypt the subset of the software environment using the OSNK, the encrypted subset being stored in a storage; and

a decryptor to decrypt the encrypted subset using the OSNK, the encrypted subset being retrieved from the storage.

41. (original) The system of claim 37 wherein the usage protector comprises:

an encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

a decryptor to decrypt the encrypted first hash value using the OSNK, the encrypted first hash value being retrieved from the storage; and

a comparator to compare the decrypted first hash value to a second hash value to generate a compared result, the compared result indicating whether the subset of the software environment has been modified.

42. (original) The system of claim 37 wherein the usage protector comprises:

a first encryptor to encrypt a first hash value of the subset of the software environment using the OSNK, the encrypted first hash value being stored in a storage;

a second encryptor to encrypt a second hash value using the OSNK; and

a comparator to compare the encrypted second hash value to the encrypted first hash value to generate a compared result, the encrypted first hash value being retrieved from the storage, the compared result indicating whether the subset of the software environment has been modified.

43. (original) The system of claim 37 wherein the usage protector comprises:

a decryptor to decrypt a protected private key to generate a private key using the OSNK;

a signature generator coupled to the decryptor to generate a signature of the subset of the software environment using the private key, the signature being stored in a storage; and

a signature verifier to verify the signature to generate a modified/not modified flag using a public key, the signature being retrieved from the storage, the modified/not modified flag indicating whether the subset of the software environment has been modified.

44. (original) The system of claim 37 wherein the usage protector comprises:

a manifest generator to generate a manifest of the subset of the software environment, the manifest describing the subset of the software environment, the manifest being stored in a storage;

a signature generator coupled to the manifest generator to generate a manifest signature of the manifest using a private key, the private key being decrypted using the OSNK, the manifest signature being stored in the storage;

a signature verifier to verify the manifest signature to generate a signature verified flag using a public key, the manifest signature being retrieved from the storage; and

a manifest verifier to verify the manifest to generate a manifest verified flag, the manifest being retrieved from the storage, the manifest verified flag and the signature verified flag being tested by a test center, the test center generating a pass/fail signal indicating whether the subset has been modified.

45. (canceled)

46. (original) The system of claim 37 wherein the software environment comprises an operating system selected from the group consisting of a Windows operating system, a Windows 95 operating system, a Windows 98 operating system, a Windows NT operating system, and a Windows 2000 operating system.

47. (previously presented) The system of claim 37 wherein the subset of the software environment comprises a registry of an operating system.

48. (currently amended) The system of ~~claim 38~~ claim 37, wherein the BK0 is generated at random on a first invocation of a processor nub.

49. (currently amended) The system of claim 37, further comprising:

a memory responsive to the processor, the memory to include an isolated memory area, the isolated memory area to be accessible to the processor in the isolated execution mode and inaccessible to the processor in the normal execution mode; and

the isolated memory area operable to receive the OS nub during a boot process.

50. (currently amended) An apparatus according to claim 1, wherein:

the platform comprises a memory responsive to the processor, the memory to include an isolated memory area, the isolated memory area to be accessible to the processor in the isolated execution mode and inaccessible to the processor in the normal execution mode; and

the isolated memory area operable to receive the OS nub during a boot process.

51. (previously presented) A method according to claim 13, wherein:

the platform comprises a memory responsive to the processor, the memory to include an isolated memory area, the isolated memory area to be accessible to the processor in the isolated execution mode and inaccessible to the processor in the normal execution mode; and

the method further comprises loading the OS nub into the isolated memory area during a boot process for the platform.

52. (currently amended) A computer program product according to claim 25, ~~wherein~~ comprising:

~~the platform comprises a memory responsive to the processor, the memory to include an isolated memory area, the isolated memory area to be accessible to the processor in the isolated execution mode and inaccessible to the processor in the normal execution mode; and~~

~~the~~ computer readable program code to load ~~loads~~ the OS nub into ~~the~~ an isolated memory area of the platform during a boot process for the platform, the isolated memory area to reside in a memory responsive to the processor, the isolated memory area to be accessible to the processor in the isolated execution mode and inaccessible to the processor in the normal execution mode.